QUINTUS

VOX SERVER PROGRAMMER'S GUIDE

Part Number: DXX-1027-01

Contents

Before You Begin

Typographical Conventions

This guide uses the following font conventions:

Font Type	What It Means
`code`	This font signifies commands or information that you enter into the computer, or information contained in a file on your computer.
italics	This font is used to add emphasis to important words and for references to other chapter names and manual titles.
jump	Blue text in online documents indicates a hypertext jump to related information. To view the related material, click on the blue text.

Notes, Tips, and Cautions

Note: A note calls attention to important information.

Tip: A tip offers additional how-to advice.

Caution: A caution points out actions that may lead to data loss or other serious problems.

Contacting Technical Support

If you are having trouble using Quintus software, you should:

1 Retry the action, carefully following the instructions given for that task in the written or online documentation.

2 Check the documentation that came with your hardware for maintenance or hardware-related issues.

3 Note the sequence of events that led to the problem and the exact messages you see. Have the Quintus documentation available.

4 If you continue to have a problem, contact Quintus Technical Support by:

- logging in to the Quintus Technical Support web site (*www.quintus.com/qq*).
- Calling (888) TECHSPT or (888) 832-4778 in the U.S. and Canada from 8:30 a.m. to 8:30 p.m. (EST), Monday through Friday (excluding holidays). International users should call 512-425-2201, or calling 1-800-242-2121 in the U.S. only.
- Email your question or problem to *support@quintus.com*. You may be asked to email one or more files to Technical Support for analysis of your application and its environment.

Note: If you have difficulty reaching Quintus Technical Support through this URL or this email address, please go to *www.avaya.com* for further information.

Product Documentation

Most Quintus product documentation is available in both printed and online form. However, some reference material is available only online and certain information is available only in printed form. A PDF document with detailed information about all of the documentation for the Quintus eContact Suite is included in the `Doc` directory on the product CD-ROM. This PDF document is also included on the separate Documentation CD-ROM.

Readme File

The Readme file is an HTML file included on the Quintus eContact Suite software CD-ROM. This file contains important information that was collected too late for inclusion in the printed documentation. The Readme file can include installation instructions, system requirements, information on new product features and enhancements, suggested workarounds to known problems, and other information critical to successfully installing and using your Quintus software. You may also receive a printed Readme Addendum, containing similar information uncovered after the manufacture of the product CD-ROM. You should review the Readme file and the Readme Addendum before you install your new Avaya software.

Electronic Documentation

The electronic documentation (in PDF and/or HTML format) for each Quintus eContact Suite product is installed automatically with the program. Electronic documentation for the entire Quintus product suite is included on the product CD-ROM and the documentation CD-ROM.

Printed Documentation

You can purchase printed copies of these manuals separately. Consult with your sales representative or with Customer Support for assistance.

License to Print the Electronic Documentation

Online copies of documentation are included on the CD for every software release. Quintus customers who have licensed software (each, a “Licensee”) are entitled to make this online documentation available on an internal network or “intranet” solely for Licensee's use for internal business purposes, and Licensees are granted the right to print the documentation corresponding to the software they have purchased solely for such purposes.

Right-To-Print License Terms

Documents must be printed “as-is” from the provided online versions. Making changes to documents is not permitted. Documents may only be printed by any employee or contractor of Licensee that has been given access to the online documentation versions solely for Licensee's internal business purposes and subject to all applicable license agreements with Quintus. Both online and printed versions of the documents may not be distributed outside of Licensee enterprise or used as part of commercial time-sharing, rental, outsourcing, or service bureau use, or to train persons other than Licensee's employees and contractors for Licensee's internal business purposes, unless previously agreed to in writing by Quintus. If Licensee reproduces copies of printed documents for Licensee's internal business purposes, then these copies should be marked “For internal use only within <Licensee> only.” on the first page or cover (where <Licensee> is the name of Licensee). Licensee must fully and faithfully reproduce any proprietary notices contained in the documentation. The copyrights to all documentation provided by Quintus are owned by Quintus and its licensors. By printing any copy of online documentation Licensee indicates its acceptance of these terms and conditions. This license only governs terms and conditions of printing online documentation. Please reference the appropriate license agreement for terms and conditions applicable to any other use, reproduction, modification, distribution or display of Quintus software and documentation.

Educational Services

Avaya University provides excellent training courses on a variety of topics. For the latest course descriptions, schedules, and online registration, you can get in touch with us:

- Through the web from *http://learning2.avaya.com*.

- Over the phone at 800-337-8941 or 800-288-LEARN (800-288-5327).
- Through email at *trainingadmin@quintus.com*.

CHAPTER 1 OVERVIEW

This chapter gives a brief, high-level overview of the VOX Server in relation to other eContact Telephony components.

The VOX Server, VRUs, and eContact Telephony

The VOX Server provides Voice Response Units (VRUs) with a connection into the world of eContact Telephony. The VRU (also known as IVR: Interactive Voice Response unit) interacts with the eContact Telephony system through the VOX Server, which plays the role of an intermediary, sending and receiving messages from both environments. Because the VOX Server can communicate with both the VRU and the eContact Telephony servers, the eContact Administration Toolkit does not need to be ported to the VRU platform. Consequently, you can interact with eContact Telephony using the VRU's own scripting language and calls to the eContact Telephony external function library designed for your specific brand of VRU.

Software running on the VRU communicates with the VOX Server through a TCP/IP connection by issuing commands written in the eContact Telephony-specific syntax described in this manual. For all practical purposes, the majority of the VOX Server's commands are thin wrappers over the functionality of other eContact Telephony servers.

Linking the VRU to eContact Telephony

Before the eContact Telephony system can interact with the VRU, the VRU has to be registered in the eContact Telephony system.

Depending on your VRU, the VOX Server connects to the VRU or the VRU connects to the VOX Server. eContact Telephony has to know the port at which this connection takes place, as well as which telephone lines the VRU services.

This configuration data is entered into the eContact Directory through eContact Manager (see "Configuration," on page 27). Once the connection between VOX Server and VRU has been established, the VRU can send commands to the VOX Server and receive responses.

Informing eContact Telephony about a New Call

The illustrations in Figures 1 and 2 are representations of the events that occur when an incoming call arrives at a PBX. Figure 1 illustrates the events that occur when eContact Telephony's QWorkflow Designer is not used. Figure 2 illustrates the events that occur when QWorkflow Designer is used. However, note that events may not, and need not, occur in the order indicated in the figures. For example, depending on network loading, the TS.IncomingCall.event may arrive at the VOX Server after the VOX.newcall request arrives at the VOX Server.

Figure 3 illustrates the events that occur when an incoming call arrives at a network VRU. Since the network VRU is "in front of" the PBX, the Telephony Server is not involved.

New Call: PBX without QWorkflow Designer

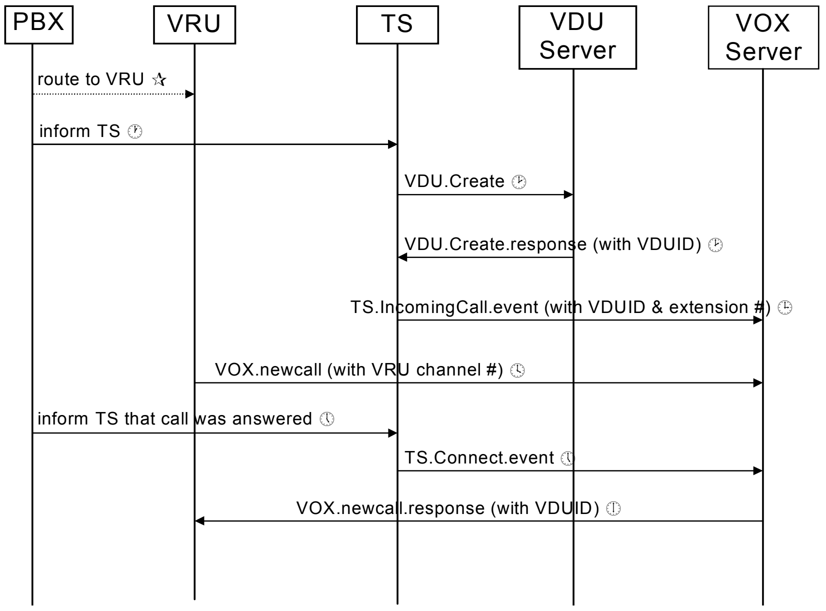
PBX
VRU
TS
VDU Server
VOX Server
route to VRU
inform TS
VDU.Create
VDU.Create.response (with VDUID)
TS.IncomingCall.event (with VDUID & extension #)
VOX.newcall (with VRU channel #)
inform TS that call was answered
TS.Connect.event
VOX.newcall.response (with VDUID)

Figure 1 Event Trace Diagram for a New Call – PBX without QWorkflow Designer

1 When a call arrives, the PBX routes the call to the VRU.

2 The Telephony Server has previously informed the PBX switch about which phone lines it is interested in monitoring. The switch has just transferred the call to one of these lines, and the Telephony Server is notified.

3 The Telephony Server then asks [**VDU.Create**] the eDU Server to create an eContact Data Unit (eDU) for the call and pass back its ID [**VDU.Create.response**]. VDUIDs uniquely identify each telephone call within the eContact Telephony system (see the *eContact Data Unit Server Programmer's Guide*). If available, Automatic Number Identifier (ANI) and Dialed Number Identification Service (DNIS) information is stored with the VDUID in the eDU.

Note: The term "eDU" refers to both the individual data repository for one call, and to the interface name of the server. It is used in both contexts throughout the documentation set. "eDU" is synonymous with "VDU," which appears sporadically in the eContact Telephony code base. "VDU" is a deprecated term, which has not yet been changed internally to "eDU" on a global basis.

4 The Telephony Server notifies [**TS.IncomingCall.event**] the VOX Server about the incoming call, including the extension to which the call was routed and the VDUID of the call.

5 Upon receiving the transferred call, the VRU informs eContact Telephony. The VRU communicates with eContact Telephony through the VOX Server, by calls to the installed library of eContact Telephony functions. The first command a VRU sends about any telephone call is the newcall request [**VOX.newcall**]. This informs the VOX Server when a telephone call has arrived on the VRU, and gives the VOX Server enough information to determine which line (channel) the call has arrived on.

6 The PBX informs the Telephony Server that the call has been picked up by the VRU, and the Telephony Server in turn informs [**TS.Connect.event**] the VOX Server. (This is the preferred process, though some PBXs cannot detect and report on connections.)

7 The VOX Server, using configuration parameters, matches up the VRU's newcall information with the incoming call event received from the Telephony Server (in particular, it matches the extension number received from the Telephony Server with the channel number received from the VRU, and associates them with the VDUID). Upon receiving the information that a connection has been made, the VOX Server returns a response [**VOX.newcall.response**], containing the VDUID, to the VRU. The VRU must either retain the VDUID and pass it to the VOX Server in all subsequent requests on behalf of the telephone call, or pass the VOX Server the appropriate channel number.

New Call: PBX with QWorkflow Designer

The following represents a common scenario. The precise use of QWorkflow Designer depends on what needs to be done and the particular PBX involved.

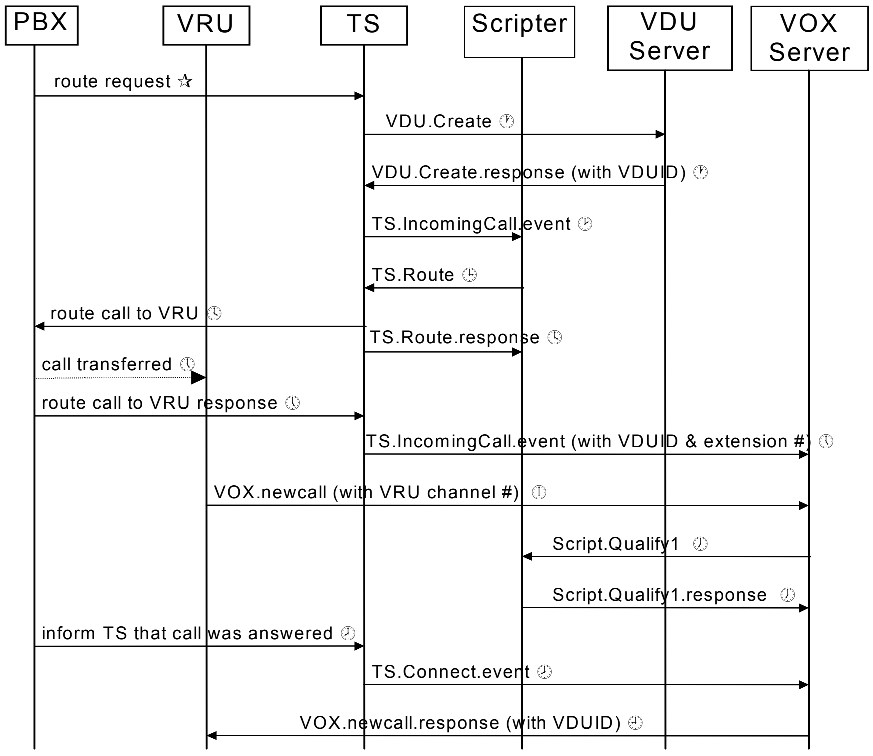
PBX
VRU
TS
Scripter
VDU Server
VOX Server
route request
VDU.Create
VDU.Create.response (with VDUID)
TS.IncomingCall.event
TS.Route
route call to VRU
TS.Route.response
call transferred
route call to VRU response
TS.IncomingCall.event (with VDUID & extension #)
VOX.newcall (with VRU channel #)
Script.Qualify1
Script.Qualify1.response
inform TS that call was answered
TS.Connect.event
VOX.newcall.response (with VDUID)

Figure 2 Event Trace Diagram for a New Call – PBX with QWorkflow Designer

1 When a call arrives, the PBX makes a routing request to the eContact Telephony Server (for a Avaya Definity switch), or CT-Connect (for Nortel Meridian Link, Nortel Symposium, Siemens and Inetcom) or executes a send data instruction from the call control table (for an Aspect switch).

 The Voice Connector Server then asks [**VDU.Create**] the eDU Server to create an eContact Data Unit (eDU) for the call and pass back its ID [**VDU.Create.response**]. VDUIDs uniquely identify each telephone call within the eContact Telephony system (see the *eContact Data Unit Server Programmer's Guide*). If available, Automatic Number Identifier (ANI) and Dialed Number Identification Service (DNIS) information is stored with the VDUID in the eDU.

 After validations with the eDU Server, the QWorkflow Server, and the Call Qualification Server, it is routed back to the switch through the Voice Connector Server and on to the VRU where automated call processing may begin.

2 Once the call has been assigned a VDUID, the Telephony Server notifies [**TS.IncomingCall.event**] the QWorkflow Designer that a new call has been received. This notification is essentially a request to the QWorkflow Designer, asking it to determine where the call should be routed.

3 For Definity and Meridian, the QWorkflow Designer replies [**TS.Route**], telling the Telephony Server to route the call to the VRU – based on the DNIS, for example. For Aspect, the QWorkflow Designer executes **TS.ReceiveData** to pass routing to the switch.

4 The Telephony Server informs the PBX, and then tells the QWorkflow Designer that it has done so. [**TS.Route.response** or **TS.ReceiveData.response**]

5 The PBX physically transfers the call to the VRU, and then tells the Telephony Server that it has made the transfer. The Telephony Server in turn informs [**TS.IncomingCall.event**] the VOX Server.

6 Upon receiving the transferred call, the VRU informs eContact Telephony. The VRU communicates with eContact Telephony through the VOX Server, by calls to the installed library of eContact Telephony functions. The first command a VRU sends about any telephone call is the newcall request [**VOX.newcall**]. This informs the VOX Server when a telephone call has arrived on the VRU, and gives the VOX Server enough information to determine which line (channel) the call has arrived on.

7 The VOX Server sends a request to the QWorkflow Designer [**Script.Qualify1**], which allows the QWorkflow Designer to fill the eDU with information about the caller, pre-fetch from databases, or take other actions. The QWorkflow Designer replies to the VOX Server [**Script.Qualify1.response**] when it is done.

8 The PBX informs the Telephony Server that the call has been picked up by the VRU, and the Telephony Server in turn informs [**TS.Connect.event**] the VOX Server. (This is the preferred process, though some PBXs cannot detect and report on connections.)

9 The VOX Server, using configuration parameters, matches up the VRU's newcall information with the incoming call event received from the Telephony Server (in particular, it matches the extension number received from the Telephony Server with the channel number received from the VRU, and associates them with the VDUID). Upon receiving the information that a connection has been made, the VOX Server then returns a response [**VOX.newcall.response**], containing the VDUID, to the VRU. The VRU must either retain the VDUID and pass it to the VOX Server in all subsequent requests on behalf of the telephone call, or pass the VOX Server the appropriate channel number.

New Call: Network VRU

For simplicity, the following represents a case where QWorkflow Designer is not used.

Note: Since the VRU is "in front of" the PBX, the Telephony Server is not involved.

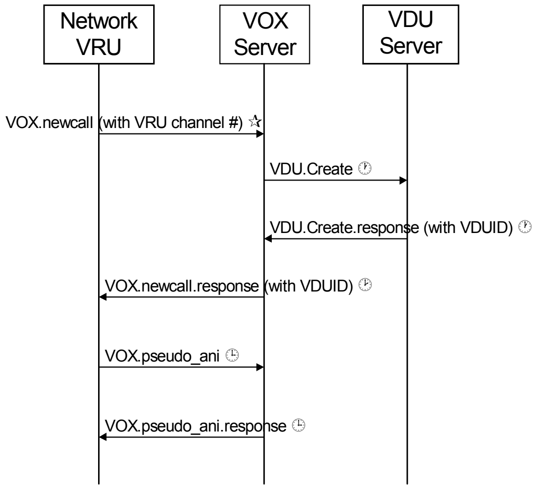
Network VRU
VOX Server
VDU Server
VOX.newcall (with VRU channel #)
VDU.Create
VDU.Create.response (with VDUID)
VOX.newcall.response (with VDUID)
VOX.pseudo_ani
VOX.pseudo_ani.response

Figure 3 Event Trace Diagram for a New Call – Network VRU

1 When a call arrives, the VRU informs eContact Telephony. The VRU communicates with eContact Telephony through the VOX Server, by calls to the installed library of eContact Telephony functions. The first command a VRU sends about any telephone call is the newcall request [**VOX.newcall**]. This informs the VOX Server when a telephone call has arrived on the VRU, and gives the VOX Server enough information to determine which line (channel) the call has arrived on.

2 The VOX Server then asks [**VDU.Create**] the eDU Server to create an eContact Data Unit (eDU) for the call and pass back its ID [**VDU.Create.response**]. VDUIDs uniquely identify each telephone call within the eContact Telephony system (see the *eContact Data Unit Server Programmer's Guide*).

3 The VOX Server associates the channel number, received from the VRU, with the VDUID. The VOX Server then returns a response [**VOX.newcall.response**], containing the VDUID, to the VRU. The VRU must either retain the VDUID and pass it to the VOX Server in all subsequent requests on behalf of the telephone call, or pass the VOX Server the appropriate channel number.

4 At some point the VRU may want to transfer the call. It would normally do this using the VDUID to identify the call. However, in the current case, the call-identifier must be passed as part of the call, and there is not enough available space in the call to pass the 32-bit VDUID. Therefore, using the VDUID, the VRU requests [**VOX.pseudo_ani**] a pseudo-ANI from the VOX Server, which the VOX Server supplies [**VOX.pseudo_ani.response**]. This pseudo-ANI is passed with the call, as though it were an ANI, and serves to identify the call.

Informing eContact Telephony to End or Transfer a Telephone Call

When the telephone call is transferred from the VRU, the VOX Server must be notified.

Note: In a system using a network VRU, or in a system without a PBX, no Telephony Server is involved with the incoming call at the VRU. Since the VOX Server command VOX.transfer makes use of the Telephony Server, this command is not available in such a system.

When the VRU no longer needs the eDU (e.g., after a hangup), the VOX.gone command is issued and the VRU cannot refer to that call's VDUID again.

Note: Flash-hook transfers by the VRU are not supported.

Other Interactions with eContact Telephony

All other interactions with the VOX Server are optional and determined by the scripts you write for the VRU.

In a typical scenario, you might want to write a script to collect touch-tone input from the caller, store it in the call's eDU, fetch information about the call or the caller from a database, and transfer the call to a human agent. To access eContact Telephony services such as data retrieval and transfers, you would incorporate calls to the eContact Telephony external function library in your VRU script. Figure 4 illustrates a typical example of the VRU communicating with the VOX Server to access eContact Telephony services.

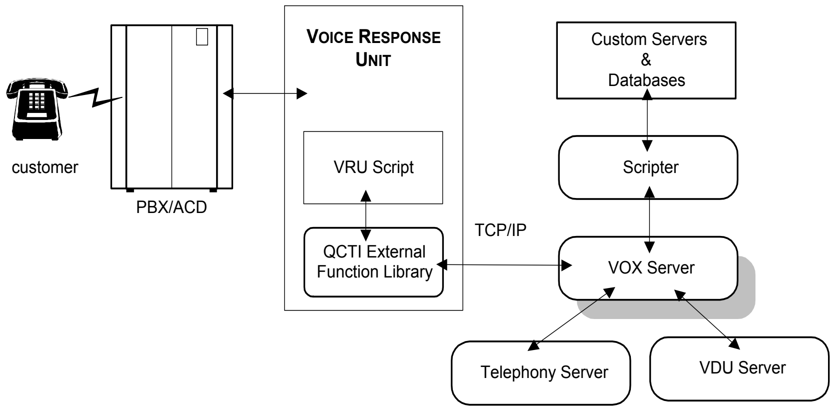

Figure 4 VRU in Relation to VOX Server

The VRU script invokes an eContact Telephony external function that is sent over a TCP/IP connection to the VOX Server, the entry point into the eContact Telephony environment. The VOX Server interprets the command, passes the request on to the appropriate eContact Telephony components, and returns the response in a form the VRU can comprehend.

eContact Telephony supports many brands of VRUs, some with their own external function library. Every vendor-specific eContact Telephony external function library comes with its own documentation.

Transferring a Call from a Network VRU

When a network VRU is ready to transfer a call, it issues a VOX.pseudo_ani request to the VOX Server (see step #4 in "New Call: Network VRU," on page 19), supplying in the request the VDUID of the call for which it wants the pseudo-ANI.

The VOX Server responds with a pseudo-ANI – taken from the list provided by the configuration parameter **pseudo-ANIs** (see "pseudo-ANIs," on page 33). The VOX Server preserves this association between the supplied pseudo-ANI and the provided VDUID – for no longer than the time indicated by the configuration parameter **pseudo-ANI Timeout** (see "pseudo-ANI Timeout," on page 34).

Within this time what will normally happen is:

- the pseudo-ANI must be placed directly in the ANI field of the call or pulsed to a cooperative network provider that will populate the ANI field with the pseudo-ANI
- the network VRU passes the call through the network and the call is delivered to the PBX
- the PBX informs its Telephony Server
- the TS extracts the pseudo-ANI from the call
- the TS requests the VOX Server to send the VDUID that is associated with the pseudo-ANI
- the VOX Server responds with the VDUID

When the VOX Server has responded, or the pseudo-ANI Timeout has passed, the given pseudo-ANI is free to be associated with a new call.

Understanding Routing in eContact Telephony

If your system includes Voice Response Units (VRUs) that use the Workflow server to route calls, you must add a Workflow UUID as an agent login id to enable routing events to go into the database. For more information, see:

- Workflow server configuration parameters section of the eContact Manager User's Guide
- Instructions for adding a UUID in the Configuration section of the QeS Installation Guide.

Add a VOX UUID

Systems with Voice Response Units (VRUs) that are using a VOX Server to route calls must add a VOX uuid as an agent login id to enable routing events to go into the database. You can find the uuid in the VOX configuration parameters sections of the VOX Programmer's Guide or the eContact Manager User's Guide. Instructions for adding a VOX UUID are provided in the Configuration section of the QeS Installation Guide.

CHAPTER 2 THE VOX SERVER VALUE CACHE

The VOX Server maintains a cache of eDU values. This reduces the volume of calls to the eDU Server, keeping response times fast on busy systems. This cache is transparent to the VRU. However, the existence of the cache does affect how you code VRU scripts: you have to decide whether the VRU issues the **VOX.getvox** command or the **VOX.getvdu** command when the VRU wants to request information about a call from eContact Telephony.

The following chart compares **VOX.getvox** and **VOX.getvdu**:

Consideration	getvox	getvdu
Response time	*	
Reduce network traffic	*	
Relatively static data	*	*
Frequently changed data		*
Retrieval of multiple values		*

The **VOX.getvox** command returns information from the VOX Server value cache and offers the advantage of speed. The command simply returns a value out of the cache and performs no checking, thus making the process a quick one. In a typical system, this method could be suitable for retrieving account numbers, ANIs (Automatic Number Identifiers), and other relatively static information unlikely to change over the course of a phone call.

The **VOX.getvdu** command returns information directly from the eDU Server, which is always the more reliable data source. Although the eDU Server can return values quickly, it is a potential system bottleneck because it handles every call in the system in real time. In busy and under-configured systems, minimizing traffic to the eDU Server may become a goal.

Any attempt to set or get information from eContact Telephony (**VOX.setvdu** or **VOX.getvdu**) updates the value stored in the cache.

Note: **VOX.getvox** is quick only if the value has been placed in the cache by a previous **VOX.getvdu** or **VOX.setvdu**. If this is not true – i.e., if the value is not in the cache – **VOX.getvox** behaves like **VOX.getvdu** and goes out to the eDU to get the value.

CHAPTER 3 CONFIGURATION

The VOX Server is configured through eContact Manager. Refer to the *eContact Manager User's Guide* for configuration instructions and generic configuration parameters. This chapter discusses configuration parameters that are specific to the VOX Server.

The only configuration parameters that must be set (all others having acceptable default settings) are those associated with eContact Manager's VRU configuration tab. In the display area of the VRU tab, click the right mouse button. In the menu that appears, use the New VRU and New Line options to initially supply the needed configuration parameters, and the Edit option to change previously entered parameters.

In the VRU configuration tab, each VRU is identified by its **VRU System Name** and (in parentheses) an automatically generated *unit number.*

The New VRU option displays the VRU Editor window. The New Line option displays the VRU Line Editor window.

There are three tables in this chapter. They present:

1. Parameters entered through the VRU Editor window
2. Parameters entered through the VRU Line Editor window
3. Parameters entered through the VOX configuration tab

Note that if a single VOX Server is used both with VRUs to which the VOX connects and with VRUs that connect to the VOX, the two kinds of VRUs can use two different ports (supplied, respectively, by the parameters **TCP/IP Port** and **VOX Listener Port**).

VRU Editor Window Parameters

Label	**Description**
VRU System Name	The network name of the VRU (e.g., `NAB3`). You must have defined a mapping between the system name and the IP address of the VRU (usually in /etc/host). [*Note*: To have the VOX Server ***ignore*** a given VRU, immediately precede the network name with a semicolon (e.g., `;NAB3`). This only applies if the VOX Server connects to the VRU. The semicolon is disregarded if used with a VRU that connects to the VOX Server.]
Initiate connection to VOX	If the VRU is of the kind that connects to the VOX Server, check this option. If the VRU is of the kind that listens for a connection from the VOX Server, leave this option unchecked. The default is unchecked.
TCP/IP Port	The port through which the VOX Server connects to the VRU. The default, `3000`, is recommended. (If the VRU is of the kind that connects to the VOX Server, ignore this parameter, and use the parameter **VOX Listener Port**, discussed in "VOX Listener Port," on page 30, to specify the port.) This port should not be used for any other purpose. If your system supports named ports ("services"), the value of this field can be the service name. However, the service name must not begin with a digit.
Ping Time	Sets the expected time (in seconds) between pings for the given VRU. If the VRU does not issue pings at least this quickly, alarms occur. (`0` disables this parameter.) This parameter is applicable only if the VRU connects to the VOX Server. The default is `0`.

VRU Line Editor Window Parameters

The VRU System Name that was entered in the VRU Editor window appears in the VRU Line Editor window as a read-only field.

Only the channels specified in the configuration parameters for a particular VOX Server are controlled by that VOX Server. This allows multiple VOX Servers to share a VRU by dividing up the channels.

Label	**Description**

Extensions	Telephone line (extension) number(s). Line numbers must be associated with channel numbers by supplying values to the pair:
	Extensions Channel
	The Telephony Server recognizes phones by their line number; the VRU recognizes them by their channel number. The correspondence must be made in order that phone calls be handled consistently.
	You can associate phone and channel numbers one at a time. If the phone numbers are sequential, you can associate multiple phone and channel numbers in a single entry. For example:
	`4100-4104 1`
	associates extension 4100 with channel 1, 4101 with channel 2, ..., and 4104 with channel 5.
	`4300 6`
	associates extension 4300 with channel 6.
	
	If the phone numbers have leading zeros, a range will preserve the zeros if both numbers have the same number of digits, e.g., 003-007.
	
	In the unusual case where some lines are controlled by a Telephony Server and some are not, an exclamation point (!) is used to indicate those lines that are <u>not</u> connected to a Telephony Server. Thus:
	`!4100-4104 1`
	`4300 6`
	would indicate that channels 1 through 5 are not controlled by a TS, whereas channel 6 is controlled by a TS. (In mixed cases such as this, the configuration parameter No Telephony Server, discussed in "No Telephony Server," on page 33, is <u>not</u> checked.)

Channel	Channel number. See the above discussion of Extensions.

VOX Configuration Tab Parameters

Label	Description
VOX Listener Port	The port at which the VOX Server is expect to listen for the VRU to connect – `3000` is recommended. (If the VRU is of the kind to which the VOX Server connects, use the parameter **TCP/IP Port**, discussed in "VRU Editor Window Parameters," on page 28, to specify the port.) `0`, the default, disables the listener.
Scripter Method	`Qualify1` (the default) causes the Script.Qualify1method to be invoked. A blank value turns off the request. As a default, when the VOX Server is first notified that a call has arrived at a VRU port, it invokes a Script.Qualify1() method to obtain any call qualification information that a Script or QWorkFlow Server might supply.. To prevent this from happening (either because the QWorkflow Server is not being used in this way, or because there is no QWorkflow Server present in your system), use this parameter with an empty string as its value. However, if you put the QWorkFlow server's alias name in the QWorkFlow Server field, just above it, for instance "QWorkFlow", then the QWorkFlow.Qualify1 method is then called. If the QWorkflow Server is present but is not being used to provide call qualification to the VOX Server, the method response contains no useful information. Shutting off the invocation saves an unnecessary request. If no QWorkflow Server is present, the method raises an error. Shutting off the invocation saves an unnecessary request and an unnecessary error message. **Troubleshooting note:** If you leave the default "Qualify1" in the "Scripter Method" (Vox server configuration, Vox Tab), no harm is really done, and there are no alarms in the "monitor", but you will see the following entry in the vox.log after every TS.IncomingCall event: t@1 - //TS.IncomingCall associated with channel 24, vduid 3accd39d00000000 0a64076f1f440002 t@1 - //Client can't locate server to satisfy request t@1 - //Request target: Script, client group: Default t@1 - //Most likely this is because the target server is t@1 - //unavailable. Maybe it can't be started by the ORBServer, t@1 - //or possibly it crashed and can't be retried yet. [2001-04-05 13:20:45.001]t@1 - Error 04000300 shack:8011 [/vob4/source/toolkit/tools/hybrid.c:410] t@1 - //Implementation not found imp=Script, group=Default, level=1

Label	**Description**
Maximum Wait Time	If an attempt by the VOX Server to connect to a VRU takes more seconds than the value entered here, the attempt is aborted (fails). Since connect attempts are done synchronously, it is useful to set this parameter for VRUs that tend to become available and unavailable frequently. If this parameter is set to `0` (the default), the VOX Server will try until a system-defined limit is reached. **Note:** This parameter has no effect if the VRU is of the kind that connects to the VOX Server.
Disable Wait	If an attempt by the VOX Server to connect to a VRU failed – and to do so took more seconds than the value entered here – future attempts to connect to that VRU are disabled. (To undo this result, see "IDL Specification," on page 37.) The default is `6`. To have the VOX Server not disable an unresponsive VRU, set this parameter to `0`. **Note:** If the VRU box is alive (i.e., the operating system is running), the VOX Server connect may succeed – even if the voice system is not running and call processing cannot be done. In such a case, this parameter has no effect. **Note:** If the value of this parameter is less than the value of the Maximum Wait Time parameter, this parameter has no effect. **Note:** If the VRU is of the kind that connects to the VOX Server, this parameter has no effect.
Wait For New Call	The time (in seconds) that the VOX Server will wait for VOX.newcall after receiving a TS.IncomingCall.event. If the designated time expires, the VOX Server raises an alarm (*Late*) and ignores the TS.IncomingCall.event. The default is `10`. If set to `0`, the VOX Server will wait until a system-defined limit is reached. For additional discussion, see "Possible Complications," on page 74.

Label	Description
Wait After Disc.	The time (in seconds) that the VOX Server will wait for VOX.newcall after receiving a TS.Disconnect.event. If the designated time expires, the VOX Server raises an alarm (*Abandon*) and ignores the TS.Disconnect.event. The default is `1`. `0` seconds is an acceptable value. This parameter will never cause the VOX Server to wait longer than the Wait For New Call parameter would allow. In general, a TS.Disconnect message that precedes a VOX.newcall should indicate that the call did not arrive at the VRU – no VOX.newcall will ever be received for that call. However, a delay in processing might cause a VOX.newcall to be emitted a few seconds late; so the VOX.newcall for a call and the TS.Disconnect that ends it can "cross". (On a system, VRU, or network that is heavily loaded – and VRU messages are subject to delays from all three – messages can be delayed by 2 seconds or more.) This parameter helps prevent misassociations from occurring if the TS.Disconnect precedes the VOX.newcall with which it should be associated. The logic is that a VOX.newcall, produced by the same telephone call that produced the TS.Disconnect, should still arrive significantly faster than a VOX.newcall that was produced by a second call coming in after an aborted call. Therefore, if an appropriate maximum time is chosen, there should be no misassociation of an aborted call's TS.Disconnect with a second call's VOX.newcall.
Wait For Incoming Call	The time (in seconds) that the VOX Server will wait for TS.IncomingCall.event after receiving a VOX.newcall request. If the designated time expires, the VOX Server raises an alarm (*Late*) and ignores the VOX.newcall request. The default is `8`. If set to `0`, the VOX Server will wait until a system-defined limit is reached. For additional discussion, see "Possible Complications," on page 74.
Show Pings	If checked, pings are included in the log. The default is unchecked.
AssignOK	If unchecked, the informational alarm AssignOK is turned off. The default is checked.

Label	**Description**
Wait for Connect	Turns off/on the requirement that the VOX Server wait for a TS.Connect.event before proceeding with call handling. The default is on (checked). If checked (on), a TS.IncomingCall.event must be followed by a TS.Connect.event before the VOX Server will proceed with call handling. If the VOX Server does not get a TS.Connect.event, a second TS.IncomingCall.event will cause it to abandon the first call (raising the alarm *NoDeliver*) and take up the second. If unchecked (off), whenever the VOX Server receives a TS.IncomingCall.event, it pretends that it has seen a following TS.Connect.event and does not wait. This is not recommended – see the discussions in "Diagnosing Problems," on page 71.
No Telephony Server	If checked, informs the VOX Server that no Telephony Server is available. The default is unchecked. If checked, this causes the VOX Server to issue a VDU.Create request upon receipt of a VOX.newcall command. Otherwise it is the Telephony Server's responsibility to request the creation of an eDU. (The usual reason that no Telephony Server is available is because a network VRU is being used, or because no PBX is being used.) In the unusual case where some lines are controlled by a TS and some are not, do not check this parameter. Instead, use the exclamation point (!) in association with the relevant Extensions parameter values – as described in "VRU Line Editor Window Parameters," on page 29.
pseudo-ANIs	*Used only with network VRUs.* A comma-separated list of pseudo-ANIs that can be used when transferring calls. The size of this list determines the maximum number of transfers involving a network VRU that can be in progress at any give time. Any element in the list may have one of the following four forms: ■ 6175551212 - The single value 6175551212. ■ `61755512XX` - The one hundred values 6175551200 through 6175551299. ■ `6175551200-6175551299` - The one hundred values 6175551200 through 6175551299. ■ `6175551200-99` - The one hundred values 6175551200 through 6175551299. Thus, a list containing the elements 6175551208,6175551209,617555121X,6175551220-4 would assign all the numbers 6175551208 through 6175551224 to be available as pseudo-ANIs. If more than one VOX Server is being used, each VOX Server must have a unique list of pseudo-ANIs.

Label	Description
pseudo-ANI Timeout	*Used only with network VRUs.* The time (in seconds) that the VOX Server will wait before breaking the association between a pseudo-ANI and an eDU. The default is `20`. The minimum useful value for this parameter is probably ten seconds. When a network VRU is ready to transfer a call, it issues a VOX.pseudo_ani request to the VOX Server (see step #4 in the example in "New Call: Network VRU," on page 19), supplying in the request the VDUID of the call for which it wants the pseudo-ANI. The VOX Server responds with a pseudo-ANI and preserves an association between the supplied pseudo-ANI and the provided VDUID – for no longer than the time indicated by this parameter. Within this time what will normally happen is: ▪ the pseudo-ANI must be placed directly in the ANI field of the call or pulsed to a cooperative network provider that will populate the ANI field with the pseudo-ANI ▪ the network VRU passes the call through the network and the call is delivered to the PBX ▪ the PBX informs its Telephony Server ▪ the TS extracts the pseudo-ANI from the call ▪ the TS requests the VOX Server to send the VDUID that is associated with the pseudo-ANI ▪ the VOX Server responds with the VDUID
Debug output	Entering a filename as the value for this parameter, initializes a special debugging facility. The debugging facility places its output in the file indicated. This output is an analysis of most of the traffic between the VRUs and the VOX Server, and describes each call received. Additional information on the debugging facility is found in "The Debugging Facility," on page 65.
Incoming/NewCall range	When the Debug output parameter is used, this defines the number of seconds that may pass between a TS.IncomingCall.event from the Telephony Server and the VOX.newcall from the VRU, before the debug code assumes that something is wrong. The maximum is `15`. Entering `0` (the default) disables this check. This parameter has no effect on the Wait For New Call parameter.

Label	**Description**
Connect/NewCall range	When the Debug output parameter is used, this defines the number of seconds that may pass between a TS.Connect.event from the Telephony Server and the VOX.newcall from the VRU, before the debug code assumes that something is wrong. The maximum is `10`. Entering `0` (the default) disables this check.
Disconnect/Gone range	When the Debug output parameter is used, this defines the number of seconds that may pass between a TS.Disconnect.event from the Telephony Server and the VOX.gone from the VRU, before the debug code assumes that something is wrong. The default is `45`. Entering `0` disables this check.

CHAPTER 4
IDL SPECIFICATION

The Interface Definition Language (IDL) is defined within the CORBA standards. It is used to create interfaces that are called by client objects and provided by object implementations. Software interfaces described by this language are both machine and language independent. eContact Telephony's IDL compiler processes the machine independent IDL definitions written by the programmer into machine-specific code.

An interface definition written in IDL completely defines the interface and fully specifies each operation's parameters. You can find information about IDL syntax and semantics in the published CORBA standards entitled *The Common Object Request Broker: Architecture and Specification.*

Below is the IDL description of the VOX Server. This interface description indicates that the VOX Server inherits the default methods from the General class (the generic server class), and has the following additional method:

```
interface VOX : General {
             void EnableVRU( in string name, in long unit_num, out long
count );
}
```

EnableVRU

IDL Syntax

```
void EnableVRU( in string name, in long unit_num, out long count );
```

Description

This method is used to cause the VOX Server to continue to attempt to connect to a VRU that it has decided is unreachable. A VRU that was determined to be unreachable shows up in the results of a Generic.GetStatus() method as discarded. eContact Manager can display this status information.

If the configuration parameter **Initiate connection to VOX** is not checked, the VOX Server will attempt to connect to the VRU when the VOX Server starts, and will attempt to reconnect if it fails. However, the **Disable Wait** configuration parameter may cause the VOX Server not to attempt any more connections after a failure. This is useful if the VOX Server serves several VRUs, in that it prevents repeated, long connect attempts to an ailing VRU from affecting service to the others. But it also means that the VOX Server will ignore the ailing VRU until the VOX Server is restarted.

To cause the VOX Server to try the connect attempt again, the EnableVRU method can be invoked. The VRU specified by the name and unit number will be flagged as in need of a connection.

The actual connection attempt is generally delayed a few seconds. This method does not wait to see if the attempt was successful. It only gives the VOX Server permission to try.

Input Parameters

Name	Description/Comments
name	This is either the **VRU System Name**, as described in "VRU Editor Window Parameters," on page 28, or a single * character to tell the VOX Server to attempt to connect to all disabled VRUs.
unit_num	This is either the automatically generated unit number of the VRU, or a 0 if * was used in *name*. The unit number is displayed (in parentheses) next to the VRU System Name, in eContact Manager's VRU configuration tab.

Output Parameters

Name	Description/Comments
count	This returns the number of VRUs affected by the request. If it is 0, *name* or *unit_num* did not correspond to a VRU in a disabled state.

Example The following is an example that reflects the way that this method would be entered using the **DDE Direct** option from the **Tools** menu in the eContact Manager.

```
[VOX.EnableVRU(″NAB3″,1,)]
```

CHAPTER 5
VRU INTERACTIONS WITH THE VOX SERVER

This chapter presents a general overview of the commands that the VRU issues, from within its scripting language, to interact with the VOX Server. These commands are the low level, general purpose functions used to communicate with VRUs that "talk" TCP/IP. Descriptions of the specific commands are given in the following chapter.

Messages between the VRU and the VOX Server

All messages between the VRU and the VOX Server are to be 1024 bytes long. The first unused byte is 0 (null) and the rest are ignored. (That is, a null byte in the message indicates that byte, and all following bytes, are to be ignored.) The maximum usable message text length is 1023 bytes – since in a message of maximum length the 1024th byte must be null.

The method by which the VOX Server and VRU establish a connection depends on the VRU. There are two methods:

The VOX Server connects to the VRU. The VRU must be started first. The VOX Server application (**voxsrv**) looks for the VRU by attempting to connect to a known TCP/IP address. The length of time that the VOX Server will spend attempting to connect to the VRU can be controlled by configuration parameters (**Maximum Wait Time** and **Disable Wait**).

The VRU connects to the VOX Server. The VOX Server must be started first.

The VOX Server is informed which method is being used by means of the parameter **Initiate connection to VOX** (see "VRU Editor Window Parameters," on page 28).

The Hello and Ping Messages

Note: The "hello", "ping", and "pong" messages are generally handled by the eContact Telephony external library. This section is provided for the information of programmers writing their own external library.

Once a connection has been established, the VOX Server immediately sends a message that consists of the text "hello" (without the quotation marks), a space, a channel number, a space, a channel number, ..., a space, the final channel number, a null byte, and n indeterminate bytes – where n is whatever number is required to make a total of 1024 bytes. (The channel number information, indicating which channels are associated with the given VOX Server, is particularly helpful if multiple VOX Servers are sharing a single VRU.) The VRU should not attempt to use the newly established connection until it has received this message.

After a connection has been established and the "hello" message sent, the VOX Server periodically sends a "ping" message (the text "ping", a null byte, and 1019 indeterminate bytes). The VRU should automatically respond with a "pong". If it does not, the VOX Server will eventually assume that the VRU has failed. (Do not confuse this "ping" message, automatically issued by the VOX Server, with the VOX.ping command issued by the VRU.)

The "hello" and "ping" messages are unique in that they are sent unsolicited from the VOX Server to the VRU. All other messages are (at this time) sent only upon VRU request. They are also special in that they do not follow the syntax of requests and responses used in all other messages.

The "hello" message serves a double purpose. It checks that the connection is in fact established. And it aids in the diagnosis of a problem that has appeared in some TCP/IP implementations which causes a broken and re-established connection to appear as if it never broke. If the VRU software sees a second "hello" message within any connection, it should assume that the connection broke and reformed, and reset any state information it has for that connection accordingly.

Requests to the VOX Server

Once the "hello" message has been received, the VRU is permitted to make requests of the VOX Server. At any time, the VRU can create a specially formatted null terminated string, store it in a buffer of length 1024, and send it to the VOX Server. Most, but not all, requests generate responses that are sent back to the VRU. The VRU should not assume that responses are received in the order implied by the requests, although that is the usual case.

To keep requests and responses straight, a request may and should contain a request ID (**reqid**) – a unique identifier generated by the VRU and echoed back in responses. This identifier can be any text of your choosing, composed of less than 15 alphanumeric characters. It should not contain leading white space. The simplest, most reliable approach is to increment a counter and use its decimal or hexadecimal expansion as the unique ID. The VOX Server does not interpret this text, so the VRU is free to encode in the text any information that would be useful to itself. For example, the string might be a hexadecimal expansion of a 32 bit number made by storing the channel number in the high byte and a unique value in the low three bytes.

If erroneous (i.e., defective or unrecognized) commands are sent, the VOX Server raises an alarm and may close the connection. However, if erroneous responses are received by the VRU, it must continue to operate, although it can choose to close the connection, raise an alarm, or do anything else that does not disturb callers.

Command Syntax

Commands are formatted as follows:

```
[commandname(arg1,arg2,...)][reqid]
```

commandname is the name of the command, and **reqid** is the unique and short alphanumeric string described in the previous section. Any arguments are dictated by the command and are quoted strings. Spaces are not allowed between arguments.

In requests that do not generate a response, the request ID can be left empty or omitted altogether.

Responses are formatted as follows:

```
[commandname.response(status,arg1,...)][reqid]
```

commandname is taken from the request, **response** is literal text, and **status** is a text string describing any error that occurred. (If no problems were detected, the string is empty.) The **args** are given in the order presented, with new results filled in. **reqid** is taken from the request.

The following example should help make this clear.

Example of a command and the response returned
[VOX.getvdu("745...3211","ani",)][117]
[VOX.getvdu.response(,"745...3211","ani","5089993244")][117]

Notice that the first response parameter is simply a comma, meaning it is empty. This indicates that no problem occurred.

The code issuing the command knows beforehand if the command will generate a response. It should be prepared to establish time-outs for missing or late responses, and to discard responses that arrive after the time-out.

VRU Conformance

VRUs must be able to interact with the VOX Server in the manner described in this chapter. In particular, VRUs must be able to properly generate all VOX commands (particularly the commands VOX.newcall and VOX.gone, described in the next chapter). Mishandled commands will almost assuredly cause the VOX Server to become confused.

CHAPTER 6
VOX SERVER COMMANDS REFERENCE

This chapter contains definitions of the VRU commands issued to the VOX Server. The commands are arranged in alphabetical order.

Function Name	Description
VOX.alarm	Raise an eContact Telephony alarm
VOX.getvdu	Fetch a value from the eDU Server
VOX.getvox	Fetch a value from the VOX Server's cache
VOX.gone	The VRU Script is done with the eDU; ready for next call
VOX.newcall	A new call arrived on the VRU; here is its line
VOX.ping	Test the VOX Server connection
VOX.pseudo_ani	Return a pseudo-ANI
VOX.setvdu	Set a value in the named eDU
VOX.tr<n><op>	Perform a transaction operation
VOX.transfer	Transfer the call

Please note the following specifics in the discussion of each function:

Command Syntax	The command format is discussed in "Command Syntax," on page 43. The trailing [reqid] indicates an optional request ID (see "Requests to the VOX Server," on page 43).
Response Syntax	The format of a response is discussed in "Command Syntax," on page 43.
Description	Briefly states the function's purpose and may offer critical information or helpful usage hints.
Errors	Possible error conditions.

Note: Any command that can accept a VDUID can also accept a string of the form #channelnum (for example: `#17`). On some VRUs this may be simpler than storing a VDUID. It also prevents problems with ambiguous VDUIDs if a call is transferred from a VRU right back to the VRU.

VOX.alarm

Syntax `[VOX.alarm(alarmname,priority,description)][reqid]`

Response `[VOX.alarm.response(,alarmname,priority,description)][reqid]`

Description Raise an eContact Telephony alarm with the given name and descriptive text.

alarmname	Name of the alarm
priority	Priority of the alarm: Low, High, Info, EMERGENCY
description	Text that describes the error

Errors Missing arguments

Example

```
[VOX.alarm("InvalidPIN","Low","Entered Invalid PIN")][234]
[VOX.alarm.response(,"InvalidPIN","Low","Entered Invalid PIN")][234]
```

VOX.getvdu

Syntax

```
[VOX.getvdu(vdu_id,name,)][reqid]
[VOX.getvdu(vdu_id,name1,,name2,)][reqid]
[VOX.getvdu(#channelnum,name,)][reqid]
[VOX.getvdu(#channelnum,name1,,name2,)][reqid]
```

Response

```
[VOX.getvdu.response(,vdu_id,name,value)][reqid]
[VOX.getvdu.response(,vdu_id,name1,value1,name2,value2)][reqid]
[VOX.getvdu.response(,#channelnum,name,value)][reqid]
[VOX.getvdu.response(,#channelnum,name1,value1,name2,value2)][reqid]
```

Description

Fetch a value related to the call. This is the correct way to get a value that might change during a call. The VOX Server's cache is bypassed and the current value in the eDU Server is fetched. The VOX Server's cache is updated with the result. If the requested value is not listed in the eDU Server, the response comes back with an empty value.

Accessing more than one element at a time is supported on some VRUs. Please check the VRU-specific documentation. The absolute upper limit is 255 name/value pairs.

The VRU may provide a # and the channel number rather than the VDUID.

vduid	VDUID of the current call
channelnum	Channel number as given in the VOX.newcall request
name	Parameter to get from the eDU
value	Value of the parameter requested

Errors

Bad vduid

Missing arguments

Example

```
[VOX.getvdu("745…3211","balance",)][234]

[VOX.getvdu.response(,"745…3211","balance","3000")][234]

[VOX.getvdu("#11","balance",)][432]

[VOX.getvdu.response(,"#11","balance","3000")][432]
```

VOX.getvox

Syntax

```
[VOX.getvox(vdu_id,name,)][reqid]
[VOX.getvox(#channelnum,name,)][reqid]
```

Response

```
[VOX.getvox.response(,vdu_id,name,value)][reqid]
[VOX.getvox.response(,#channelnum,name,value)][reqid]
```

Description Fetch a value related to the call. The VOX Server looks up the value in its eDU value cache. If it can find it, the value is returned to the VRU. Otherwise, the VOX Server asks the eDU Server to look it up and return the result. If it is not listed in the eDU Server, the response comes back with an empty value.

This call trades off accuracy for speed. It is faster to retrieve information stored in the VOX Server's eDU value cache. However, some values may have changed since the eDU was stored in the VOX Server. Therefore, use this call to retrieve relatively static values that stay the same during any given call, such as an account number.

Note: VOX.getvox is quick only if the value has been placed in the cache by a previous VOX.getvdu or VOX.setvdu. If the value is not in the cache, VOX.getvox behaves like VOX.getvdu and goes out to the eDU to get the value. Also, VOX.getvox cannot retrieve multiple values, as VOX.getvdu can.

The VRU may provide a # and the channel number rather than the VDUID.

vduid	VDUID of the current call
channelnum	Channel number as given in the VOX.newcall request
name	Parameter to get from the eDU
value	Value of the parameter requested

Errors Bad vduid

Missing arguments

Example

```
[VOX.getvox("745…3211","language",)][234]

[VOX.getvox.response(,"745…3211","language","English")][234]

[VOX.getvox("#11","language",)][432]

[VOX.getvox.response(,"#11","language","English")][432]
```

VOX.gone

Syntax

```
[VOX.gone(vdu_id)][reqid]
[VOX.gone(#channelnum)][reqid]
```

Response

```
[VOX.gone.response(,vdu_id)][reqid]
[VOX.gone.response(,#channelnum)][reqid]
```

Description Indicates that the call has left the VRU. Causes the VOX Server to issue a VDU.Terminate(). The eDU is then terminated and the VOX Server can no longer access the eDU.

Note: If you have QRepository, when all parties terminate the eDU, the data in the eDU is written to QRepository. If you do not have QRepository, the data in the eDU will be lost after termination, unless you have captured this data in some way. (QRepository is discussed in the *QDecision User's Guide*.)

A VOX.gone *must* be issued when the VRU is finished with a call. There must be one VOX.gone for each VOX.newcall.

The VRU may provide a # and the channel number rather than the VDUID.

vduid	VDUID of the current call
channelnum	Channel number as given in the VOX.newcall request

Errors Bad vduid

Missing arguments

Example

```
[VOX.gone("745…3211")][234]

[VOX.gone.response(,"745…3211")][234]

[VOX.gone("#11")][432]

[VOX.gone.response(,"#11")][432]
```

VOX.newcall

Syntax

```
[VOX.newcall(channelnum,)][reqid]
[VOX.newcall(#channelnum,)][reqid]
```

Response

```
[VOX.newcall.response(,channelnum,vdu_id)][reqid]
[VOX.newcall.response(,#channelnum,vdu_id)][reqid]
```

Description A VOX.newcall is sent when the VRU is aware of a new telephone call entering its domain. It sends a string indicating which line has the call (this string is called the *channel number*).

The VOX Server returns the VDUID issued to the call. The VOX Server has to manage a three-way handshake between the VRU, the Telephony Server if present), and the QWorkflow Designer (if present) before it can return. Once it returns, the VRU can issue VOX.getvdu or VOX.getvox requests to learn what is known about the call.

The # in the command syntax is optional. It is allowed just for the sake of consistency with your other VOX Server commands if they are using the channel number rather than the VDUID.

Note: For a detailed discussion of the response of the VOX Server to an incoming call, see "Diagnosing Problems," on page 71. Also see the discussion of "Wait for Connect," on page 33, and the examples in "Informing eContact Telephony about a New Call," on page 14.

vduid	VDUID of the current call
channelnum	Channel number as given in the VOX.newcall request

Errors No channel number given

No such channel

Channel not currently controlled by eContact Telephony (usually indicates either that the mapping of channel number to line number is wrong, or that the Telephony Server is down)

Example

```
[VOX.newcall("1",)][234]

[VOX.newcall.response(,"1","745…3211")][234]

[VOX.newcall("#11",)][432]

[VOX.newcall.response(,"#11","845…3211")][432]
```

VOX.ping

Syntax `[VOX.ping()][reqid]`

Response `[VOX.ping.response(,)][reqid]`

Description Test to see if the VOX Server is alive and connected. Useful for debugging.

Note: The "ping" message referred to in "The Hello and Ping Messages," on page 42 is automatically generated by the VOX Server to reassure it that the VRU is alive and well. The current command is used by the VRU to determine whether the VOX Server is alive and well.

Errors None (since receiving or not receiving VOX.ping.response determines the success or failure of the command)

Example

```
[VOX.ping()][234]

[VOX.ping.response(,)][234]
```

VOX.pseudo_ani

Syntax

```
[VOX.pseudo_ani(vdu_id,)]
[VOX.pseudo_ani(#channelnum,)]
```

Response

```
[VOX.pseudo_ani.response(,vdu_id,ps_ani)]
[VOX.pseudo_ani.response(,#channelnum,ps_ani)]
```

Description Return a pseudo-ANI to the network VRU. (For a discussion of the pseudo-ANI, see "New Call: Network VRU," on page 19 and "Transferring a Call from a Network VRU," on page 22.)

vduid	VDUID of the current call
channelnum	Channel number as given in the VOX.newcall request
ps_ani	The pseudo-ANI that is associated with the current call

If no pseudo-ANIs are available, the request fails.

Errors Bad vduid

Missing arguments

Failed pseudo-ani

Example

```
[VOX.pseudo_ani("745…3211",)]

[VOX.pseudo_ani.response(,"745…3211","6175551212")]

[VOX.pseudo_ani("#11",)]

[VOX.pseudo_ani.response(,"#11","6175551212")]
```

VOX.setvdu

Syntax

```
[VOX.setvdu(vdu_id,name,value)][reqid]
[VOX.setvdu(vdu_id,name1,value1,name2,value2)][reqid]
[VOX.setvdu(#channelnum,name,value)][reqid]
[VOX.setvdu(#channelnum,name1,value1,name2,value2)][reqid]
```

Response

```
[VOX.setvdu.response(,vdu_id,name,value)][reqid]
[VOX.setvdu.response(,vdu_id,name1,value1,name2,value2)][reqid]
[VOX.setvdu.response(,#channelnum,name,value)][reqid]
[VOX.setvdu.response(,#channelnum,name1,value1,name2,value2)][reqid]
```

Description Set or change a value in the named eDU. The value is immediately set in the eDU. The VOX Server's eDU value cache is updated with the name and value as well.

Setting more than one element at a time is supported on some VRUs. Please check the VRU-specific documentation. The absolute upper limit is 255 name/value pairs.

The VRU may provide a # and the channel number rather than the VDUID.

vduid	VDUID of the current call
channelnum	Channel number as given in the VOX.newcall request
name	Parameter to set in the eDU and cache
value	Value to which the parameter is to be set

Errors Bad vduid

Bad argument count

Example

```
[VOX.setvdu("745…3211","balance","3000")][234]

[VOX.setvdu.response(,"745…3211","balance","3000")][234]

[VOX.setvdu("#11","balance","3000")][432]
```

```
[VOX.setvdu.response(,"#11","balance","3000")][432]
```

VOX.tr<n><op>

Syntax

```
[VOX.tr<n><op>(vdu_id,args...)][reqid]
[VOX.tr<n><op>(#channelnum,args...)][reqid]
```

Response

```
[VOX.tr<n><op>.response(,vdu_id,args...)][reqid]
[VOX.tr<n><op>.response(,#channelnum,args...)][reqid]
```

Description These commands are issued to conduct *transaction operations* on the eContact Telephony QWorkflow Designer. eContact Telephony QWorkflow Designer features are customized for each customer. Please consult the *Workflow Design Guide* for information about the QWorkflow Designer's general operating principles.

The VRU may provide a # and the channel number rather than the VDUID

tr	Literal text
<n>	The digit 1 or 2 1 - transaction that does not produce output 2 - transaction that does produce output
<op>	A short (8 characters or less) operation name
vduid	VDUID of the current call
channelnum	Channel number as given in the VOX.newcall request
args	Arguments associated with transaction

This operation accesses the customer-specific QWorkflow Designer transaction methods.

The general tr1 transaction has up to four input arguments but no output. When a tr1 type command is issued, the VOX Server translates this to a Trans1 method call to the QWorkflow Designer, with the operation name as an argument.

The general tr2 transaction has up to four input arguments, as well as output. When a tr2 type command is issued, the VOX Server translates this to a Trans2 method call to the QWorkflow Designer, with the operation name as an argument. In the case of a Trans2 method the QWorkflow Designer returns a sequence of couples back to the VOX Server, which places the results in the VOX Server's eDU cache.

Errors Bad vduid

Transaction dependent

Example

```
[VOX.tr2setaccnt("745...3211","654321")][234]
```

where 745...3211 is the VDUID, 654321 is an account number, 234 is the request ID

VOX.transfer

Syntax

```
[VOX.transfer(vdu_id,destination)][reqid]
[VOX.transfer(#channelnum,destination)][reqid]
```

Response

```
[VOX.transfer.response(,vdu_id,destination)][reqid]
[VOX.transfer.response(,#channelnum,destination)][reqid]
```

Description Transfer the call to a new destination. This calls the TS.TransferVDU() method allowing a coordinated voice/data transfer.

Note: Since VOX.transfer involves the Telephony Server, it is not available when no Telephony Server is available – as when an incoming call arrives at a network VRU.

This does not terminate the eDU. The function VOX.gone must be called to terminate the eDU. Until VOX.gone is called, the VRU script continues to access the eDU and cache.

The VRU may provide a # and the channel number rather than the VDUID.

vduid	VDUID of the current call
channelnum	Channel number as given in the VOX.newcall request
destination	Where to send the call. This can be a number or a login_id.

Note: Flash-hook transfers by the VRU are not supported.

Errors

Bad vduid

Missing arguments

Example

```
[VOX.transfer("745...3211","4001")][234]

[VOX.transfer.response(,"745...3211","4001")][234]

[VOX.transfer("845...3211","BobT")][243]

[VOX.transfer.response(,"845...3211","BobT")][243]

[VOX.transfer("#11","4001")][432]

[VOX.transfer.response(,"#11","4001")][432]
```

CHAPTER 7
ALARMS

The eContact Manager application provides system administration tools for monitoring alarm events. Visual and/or auditory alarms (beeps) trigger whenever the system detects problems that require fixing by human intervention, e.g., server failures. (For more information about monitoring alarms, see the *eContact Manager User's Manual*.)

This chapter describes the alarms associated specifically with the VOX Server. The following table contains the alarm name, its priority response category (emergency, high, low, or info), a description of the alarm, and a recommended response.

Alarm Name	Priority	Description	Cause/Recommended Action
Abandon	low	VOX.newcall is late, clearing state.	A VOX.newcall might be unacceptably delayed in relation to a TS.Disconnect.event because of network overloading or an overburdened VRU. (See "Wait After Disc.," on page 32)

Alarm Name	Priority	Description	Cause/Recommended Action
AmbiVDUID	high	A VDUID matches multiple channels.	A VDUID arrived that identifies more than one channel. This should not occur; it represents either a serious VOX Server error or missing VOX.gones coupled with other errors. Call mishandling will probably occur. If a VRU transfers a call to itself, the VOX Server may interpret the call as being on two channels at once, leading to this alarm. VRU scripts that do this self-transfer should use *#channelnum* in VOX Server commands (not *vdu_id*), in order to avoid ambiguity.
AssignFail	high	Assign to line failed, will retry periodically.	The VOX Server could not get control of the line listed for a given channel from the Telephony Server. This usually means some error in how the lines and channels are configured in the VOX Server, but can also mean the Telephony Server is unavailable. The VOX Server will try again.
AssignOK	info	Assign succeeded for the specified line.	The VOX Server succeeded in getting control of the line. If it only succeeded after a number of failures, there may be Telephony Server problems, and log files should be collected and checked. (See "AssignOK," on page 32.)
BadAddr	emergency	A bad port or an unknown IP address has been identified.	The VOX Server could not make sense of the IP or port address listed for the VRU in the configuration. The TCP host definitions (check /etc/hosts) may not list the VRU, or you may have mistyped the name. No connection to the VRU can occur.

Alarm Name	Priority	Description	Cause/Recommended Action
Congest	high	Traffic to VRU has stopped.	Attempts to send messages to the VRU are not succeeding; the VRU or network is probably in a hung state. The VOX Server will keep trying, but if the situation does not clear up will eventually raise another alarm and break the connection.
ExtraVRUCon	high	A VRU has connected to the VOX Server, though the VOX Server believed that that VRU was already connected.	This usually indicates a faulty TCP/IP stack on either the VRU or the VOX Server's host. The VOX Server will close the old connection and accept the new one, causing possible interruption of service for existing calls.
Garbage	high	The VRU sent a request that could not be parsed properly; the request was discarded.	The VRU sent something that was not a recognizable request. This indicates coding errors, or memory corruption, in the VRU code or the VOX Server. Calls are probably being mishandled. Stop and restart the VRU application. If the problem persists, check for coding errors.
Late	high	VOX.newcall / TS.IncomingCall is late, clearing state.	A VOX.newcall might be missing because a call arrived at the PBX but was hung up before the VRU could deal with it, or because the VRU script does not reliably respond to a received call by sending VOX.newcall, or because a channel and a line are misassociated. (See "Wait For New Call," on page 31.) A TS.IncomingCall.event might be missing because the value for **Wait For New Call** was set too low and therefore a TS.IncomingCall.event was discarded too soon. (See "Wait For Incoming Call," on page 32.)

Alarm Name	Priority	Description	Cause/Recommended Action
LoseTS	low	The Telephony Server has failed.	The VOX Server immediately attempts to bring the Telephony Server back on-line. This can trigger additional alarms. Newly arrived or arriving calls may be mishandled, but the eContact Telephony system should quickly recover.
Lost	high	Lost newcall request.	The VOX Server expected to be answering a VOX.newcall request, but could not find the request when the time came. This should not occur. Contact customer service.
LostScriptSync	high	Unexpected QWorkflow Designer response.	A QWorkflow Designer response occurred when the VOX Server did not expect one. This might rarely occur if the QWorkflow Designer or Telephony Server is restarted, but otherwise indicates a VOX Server error.
LostVDUIDSync	high	QWorkflow Designer 's VDUID is not ours.	A QWorkflow Designer response mentions a VDUID that is not listed for the associated channel. This may occur if the VRU, Telephony Server, or QWorkflow Designer has recently been restarted, but otherwise indicates a VOX Server error.
Miss	high	TS/Newcall seen.	This almost always indicates a missing VOX.gone, or possibly a VRU that failed to hold a channel busy until it got a VOX.gone.response. A TS.IncomingCall.event/VOX.newcall announced a call where the VOX Server thought one was already in progress. The VOX Server will discard the previous call information and begin to handle the new call.

Alarm Name	Priority	Description	Cause/Recommended Action
MissInc	high	Connect without IncomingCall.	A TS.Connect.event was received with no preceding TS.IncomingCall.event. This generally means that the VOX Server discarded a TS.IncomingCall.event due to a timer.
NoAnswer	emergency	Cannot open TCP/IP connect to VRU, will retry periodically.	The VRU and eContact Telephony are not talking; calls handled by the VRU, if it is functioning, are certainly being mishandled. Check to see why the VRU is not running or is not visible to the network. The VOX Server silently retries the connection attempt every few seconds until it succeeds.
NoAssign	emergency	The assign has failed repeatedly.	The VOX Server could not get control of the line listed for a given channel from the Telephony Server. This usually means some error in how the lines and channels are configured in the VOX Server, but can also mean the Telephony Server is unavailable. The VOX Server stopped trying and sent this alarm.
NoChnList	emergency	Empty channel/line list for VRU.	The configuration information implies that the VRU should be in service, but lists no line/channel mappings. Calls are certainly being mishandled; expect other alarms to occur.
NoDeliver	info	Last call did not reach VRU.	Indicates that a call was routed to the VRU, but did not arrive (the caller probably hung up), and then a second call was routed to the VRU. The VOX Server raises this alarm and then uses the second call's information in place of the first.
NoPing	emergency	No requests from VRU; dropping connection.	A VRU has fallen silent (is not sending requests or ping messages). It will be considered hung, and disconnected.
NoPoint	emergency	No VRUs declared in configuration.	The VOX Server exits; it can do no useful work.

Alarm Name	Priority	Description	Cause/Recommended Action
NoReply	emergency	No response from VRU; dropping connection.	A VRU has fallen silent (is not responding to ping messages). It will be considered hung, and disconnected.
Off	low	VRU will be ignored.	The listed VRU has been configured for "Ignore". It will not be connected to. This is an informational message, reminding you that you configured the VRU this way.
UnkChan	emergency	The VRU indicated a new call on a channel that the VOX Server has not been told about. (It cannot determine what line the channel represents.)	Either the VRU is sending bad requests, or the VOX Server has not been told about all of the channel/line mappings. The VOX Server drops the request; it is certain the call is being mishandled. Check the channel/line mappings; check the newcall request sent by the VRU.
UnkVRU	emergency	VRU is not defined in configuration.	A VRU name is listed that was not defined in the configuration. (The alarm message may identify the variable that had the error.) Fix the configuration and restart the server.
UnkVRUCon	high	Connect attempt from a system not on the list of VRUs.	eContact Telephony does not know about this VRU; the connect is rejected. Edit the configuration data discussed in "VRU Editor Window Parameters," on page 28. Make sure that you have not confused the eContact Telephony ports (usually starting at 8000) and the VRU ports (usually starting at 3000) in the configuration.
Vanished	emergency	VRU server disconnect.	The connection to the VRU has been broken; call mishandling will occur. It almost always means that the software running on the VRU has crashed.

Alarm Name	Priority	Description	Cause/Recommended Action
VoxIllegalVDUID	high	Badly formed VDUID.	A badly formed VDUID was seen from the VRU or Telephony Server. This always represents a bug in the application that sent the offending request or event. Note the time and check the VOX log, to see what software generated the bad VDUID. This error will almost always cause other alarms to occur.
Voxini	emergency	No configuration data available.	The VOX Server could not find any configuration data. It will exit, as it cannot do anything useful without line mappings. Supply the configuration data in eContact Manager.
VoxUnknownVDUID	high	Unknown VDUID was seen.	A VDUID arrived that is not associated with any current call. This generally indicates a missing VOX.newcall, or VRU confusion. Can also indicate VOX.gone having been issued twice for the same VDUID. In a network VRU environment, it can also be caused by a request to associate a VDUID with a pseudo-ANI that has timed out.
VruDisable	emergency	Connect attempt to a VRU timed out, disabling the VRU.	A VRU did not respond to a connection attempt; the VOX Server will pretend that it does not exist. The VRU may be down, misconfigured, or the eContact Telephony software on the VRU may not be running. The VOX log file may provide more information. You may want to alter the **Disable Wait** configuration parameter and restart the VOX Server.
VruUp	low	VRU is now visible.	The VOX Server has connected to the named VRU. This is a status report — no action is required.
VruUpI	low	VRU is now visible.	The named VRU has connected to the VOX Server. This is a status report — no action is required.

CHAPTER 8 TROUBLESHOOTING

This chapter discusses the VOX Server debugging facility, the status information available for the VOX Server, the error messages that the VOX Server can return to the VRU, and tips on how to diagnose problems in integrating VRUs with eContact Telephony.

The Debugging Facility

The VOX Server contains a debugging facility that is an additional log file designed to describe individual calls. To use the debugging facility, a filename must be entered into the **Debug output** configuration parameter (see "Debug output," on page 34). Certain events in the life of a call are recorded, and upon completion of the call its history is written to the debugging log file as a single unit.

The debug log does not reflect all traffic, that information is found in VOX.log, but it does mention the main operations (newcall, gone, and setvdu). The advantage of the debug log file is that it organizes events and requests into groups by channel. The standard VOX Server log lists events and requests in the order they were received, across all channels.

A simple call description from the debug log file appears in the example below.

[state 1] indicates that the Telephony Server told the VOX Server about the call
[state 2] indicates that the VRU told the VOX Server about the call
[state 4] indicates that the QWorkflow Designer has responded
[state 8] indicates that the call was connected
[state 0] indicates that the call has ended

These values can be ORed together, e.g., [state 3] indicates that both the Telephony Server and the VRU told the VOX Server about the call.

Example

```
Call history [31b85c9200000000780000de219b0002] on Channel 1 (Line 4101)
-on Vru periphonics:0
-This call's end: Normal call end
-Duration: 12:45:06 ... 12:46:49  (103 seconds)
-
-12:45:06 TS->VOX[state 1]
-TS.IncomingCall
-IncomingCall event introduces a new call
-
-12:45:09 VRU->VOX[state 2]
-VOX.newcall
-Vru indicates newcall on channel 1
-
-12:45:09 TS->VOX[state 8]
-TS.Connect
-Connect indicates call reached vru
-
-12:45:09 VOX->VRU[state 3]
-[VOX.newcall.response(,"1","31b85c9200000000780000de219b0002")][1]
-VRU and TS in sync; goodness prevails
-
-12:45:09 TS->VOX
-TS.Connect
-Event received
-
-12:45:10 VOX->VRU
-
[VOX.getvox.response(,"31b85c9200000000780000de219b0002","account","")][
1]
-Fetch of value done
-
-12:46:36 VOX->VRU
-[VOX.setvdu.response(,"31b85c9200000000780000de219b0002","balance",
"000000000000000000000000000050099")][1]
-SetValues OK
-
-12:46:36 VOX->VRU
[VOX.getvdu.response(,"31b85c9200000000780000de219b0002","balance",
"000000000000000000000000000050099")][1]
-Fetch of value done
-
-12:46:49 VRU->VOX[state 0]
-[VOX.gone.response(,"31b85c9200000000780000de219b0002")][1]
-VOX.gone received, goodness prevails
```

Debug Configuration Parameters

The configuration parameters for the debugging facility are given and discussed at the end of "VOX Configuration Tab Parameters," on page 30.

If no filename is entered into the **Debug output** configuration parameter (see "Debug output," on page 34), the debugging facility is not used. The VOX Server does not behave differently (except for writing the log file) if the debugging facility is used.

For Hammer IT testing, in which many short calls are delivered to the VOX Server, you may want to use the values noted in the chart below:

Debug output	voxcall.log
Incoming/NewCall range	6
Connect/NewCall range	4
Disconnect/Gone range	8

Note: If a VOX.gone is severely delayed or missing, the VRU has allowed the call to end, and another call comes in on that line, a "preempt" occurs. This raises a VOX Miss alarm and also causes comments in the debug log. Preempts should not cause a call to be lost. If VOX.gone arrives, it performs as expected (terminates the old eDU). However, other functions (VOX.setvdu, etc.) raise "bad eDU" alarms if the previous eDU is accessed. This is because the VOX Server (except for the debug code) no longer "sees" the old eDU once a line has been preempted by a new call. VRU coders should be careful to cause the line to remain busy until they get a response from the VOX.gone command.

Server Status

VOX Server status information can be obtained through eContact Manager (see the *eContact Manager User's Guide*). In addition to generic server information, the following specific information is provided for the VOX Server.

Total Number of Calls

You might see information of the following kind:

```
totalcalls      132617
```

This indicates that 132617 calls have been handled by the VOX Server since startup. If the number of calls handled is 0, the line does not appear.

Total Number of VRUs

You might see information of the following kind:

```
vrus       7
```

This indicates that the VOX Server has been configured to know about 7 VRUs. If the number of VRUs is 0, the line does not appear.

Information for a Given VRU

You might see information of the following kind:

```
vru.2      <- NAB3:1     [35466 calls]
```

This indicates that the VRU, with the arbitrary sequence number 2, has been connected to by the VOX Server (<-), has the VRU System Name NAB3, has the the automatically generated unit number of 1, and has handled 35466 calls through this VOX Server.

The general format for such lines is:

```
vru.N      direc name:num {Off} {Disabled} [m calls]
```

where

N	is a sequence number, assigned in order to distinguish each VRU
direc	indicates whether the VOX Server connected to the VRU (<-), or the VRU connects to the VOX Server (->)
name	is the **VRU System Name**, as described in "VRU Editor Window Parameters," on page 28.
num	is the automatically generated unit number of the VRU. The unit number is displayed (in parentheses) next to the VRU System Name, in eContact Manager's VRU configuration tab.
m	is the number of calls handled by the VRU through this VOX Server
off	if it appears, means that no connection has been established between the VOX Server and this VRU
Disabled	if it appears, means that the VOX Server is no longer attempting to connect to this VRU

Information for a Given Channel of a Given VRU

You might see information of the following kind:

```
vru.0.channel.1.alarms8
vru.0.channel.1.calls714
vru.0.channel.1.state4100(C)745...3211
vru.0.channel.2.calls699
vru.0.channel.2.state4101(idle)
```

This indicates that:

Channel 1 of VRU 0 has currently raised 8 alarms, has currently received 714 calls, is associated with extension (line) number 4100, has most recently received a TS.Connect.event, and is dealing with the call that has the VDUID 745…3211.

Channel 2 of VRU 0 has currently raised 0 alarms (and therefore the line dealing with alarms is not displayed), has currently received 699 calls, is associated with the extension number 4101, and is currently idle.

The general format of these lines is:

vru.*N*.channel.*M*.alarms	A
vru.*N*.channel.*M*.calls	C
vru.*N*.channel.*M*.state	L (S) I

where:

N	is the sequence number discussed in the previous section
M	is the channel number
A	is the number of alarms (if 0, the line is not displayed)
C	is the number of calls (if 0, the line is not displayed)
L	is the extension (line) number
I	is the VDUID (blank if unknown)
S	indicates the "state" of the channel by some combination of five symbols:

T - TS.IncomingCall.event seen

D - VDU.Create.response seen (only used if no TS available)

C - TS.Connect.event seen or assumed (assumed if no TS, or if PBX of a kind unable to detect and report on connections)

S - Scripter response seen, or Scripter disabled

V - VOX.newcall seen

idle - the channel is currently idle

Error Messages

The various error messages that can be returned to the VRU by the VOX commands are given under each command in "VOX Server Commands Reference," on page 45. In addition, error messages can be passed on from the Telephony Server, the eDU Server, and QWorkflow Designer.

Diagnosing Problems

Note: When no Telephony Server is available – as when an incoming call arrives at a network VRU – most of the following remarks do not apply. However, the comments on a "Missing VOX.gone Request," on page 73, and "Failure of Telephony Server Request Involving a Pseudo-ANI," on page 74, together with "Suggestions on Structuring VRU Scripts," on page 77, are relevant.

In order to diagnose problems involving the integration of VRUs with eContact Telephony, it is necessary to understand how this integration is intended to work.

The VOX Server acts as a coordinator. It coordinates calls made by the Telephony Server and calls made by the VRU. This coordination process is the most important job of the VOX Server. If it fails to do this job, calls will be *misassociated*: VRU applications will be handling one call but will be presented with eContact Telephony information about another call. It is therefore important to take VOX Server alarms seriously, and to understand what is happening.

How It Should Work

The VOX Server provides a call's VDUID (the "handle" that eContact Telephony uses to uniquely identify a given call) to the VRU, and then allows the VRU script to use that VDUID to access and change the call information.

Before providing the VDUID to the VRU, the VOX Server must itself get the VDUID. It does this by telling the Telephony Server to send events to it for all the lines that lead to the given VRU. When a call comes in on one of those lines, the Telephony Server tells the VOX Server the call's VDUID by means of the TS.IncomingCall.event (see the figures and discussion in "Informing eContact Telephony about a New Call," on page 14).

The TS.IncomingCall.event not only informs the VOX Server of the call's VDUID, it also tells the VOX Server which extension (line) the call came in on. Using the mapping between channel and line in the VOX Server's configuration (see the discussion of "VRU Line Editor Window Parameters," on page 29), the VOX Server can determine which VRU channel the call is going to. (At this point the VOX Server may also have started a QWorkflow Designer request for that call, to initiate custom call handling.)

Meanwhile, the call arrives at the VRU. The VRU should immediately respond by picking up the call, and making a VOX.newcall request to the VOX Server. (The VOX.newcall request contains the call's channel number.) Since the channel number sent by the VRU corresponds to the channel number determined from the TS.IncomingCall.event from the Telephony Server, the VOX Server associates them. It then responds to the VOX.newcall with a VOX.newcall.response that contains the VDUID of the call. (By default – see the discussion of "Wait for Connect," on page 33 – the VOX Server waits for a TS.Connect.event from the Telephony Server before responding to the VOX.newcall request. If the VOX Server made a request of the QWorkflow Designer, it also waits for the QWorkflow Designer to respond before itself responding to the VOX.newcall request.)

Because networking delays can slow transmission, the VOX Server will accept the VOX.newcall and TS.IncomingCall.event in either order – it simply waits until it gets both before it responds.

When the VRU is done with the call, it must first send a VOX.gone request to the VOX Server, wait for the response, and then make the channel available for another call. (*The order matters.*) This enables the VOX Server to begin afresh on that channel, again waiting for a VOX.newcall and TS.IncomingCall.event pair for the next call.

What Can Go Wrong

Incorrect Association of Channel Numbers and Extension (Line) Numbers

The VOX Server will see VOX.newcall requests for one channel and TS.IncomingCall.events for another channel, and never associate them. Eventually other calls will occur on these channels and the VOX Server will misassociate calls, leading to chaos.

Either fix the configuration information for the VOX Server, or make the phone cables match the existing configuration.

Missing VOX.newcall Request

The VOX Server will get a TS.IncomingCall.event, wait for the VOX.newcall request, and not get it. If a second call should go to the same channel, and the VOX.newcall for the second call arrives at the VOX Server ahead of the new TS.IncomingCall.event, the VOX Server will associate the VOX.newcall with the old TS.IncomingCall.event it had already received – and misassociation occurs. This sort of error can keep a channel out of sync, with continual misassociations occurring, *even if only one VOX.newcall was dropped*.

VRU scripts should start with a VOX.newcall request *as their very first step* after picking up the call. It is tempting to put some speech steps in first, to give the VOX.newcall some cover time, but the caller may hang up at that point, causing many scripts to abort or otherwise fail to send the expected VOX.newcall. eContact Telephony delivered the call to the VRU and the VRU is expected to provide the VOX.newcall for that call, or risk misassociating all future calls on that channel. Even if the caller hangs up, the VOX.newcall (and the associated VOX.gone) for that call must be sent.

Note: It is, in any case, unnecessary to give the VOX.newcall request cover time. This is a fast operation.

Missing VOX.gone Request

In this case the VOX Server never knows that the VRU has finished with a call, and it never releases the eDU for that call. When another call arrives on that channel, the TS.IncomingCall.event or VOX.newcall gives it a hint that the last call must have ended, because a new call is arriving – and at that point it will act as if it had seen a VOX.gone request. However, this approach leaves the software open to race conditions – i.e., the state that the system thinks it is in may be different from the actual state – and is not supported. (A *race condition* exists when two competing events “race” and the result is determined by which comes in first. In such a situation an external event may occur that temporarily invalidates an application’s state information.)

Whether the caller or the VRU decides to end the call, the VRU must issue a VOX.gone, and receive a response, before it makes the channel available for another call. (Typically, VOX.gone.response returns in a small fraction of a second. Waiting for it does not impact call handling rates.)

Failure of Telephony Server Request Involving a Pseudo-ANI

Can occur only with network VRUs.
The Telephony Server receives a call containing what the TS recognizes as a pseudo-ANI. The TS requests, from the VOX Server, the VDUID corresponding to the pseudo-ANI. However, the time set by the configuration parameter **pseudo-ANI Timeout** has already passed, and therefore the VOX Server has no VDUID to return to the TS.

In such a case, the TS generates a Problem alarm, and then requests the eDU Server to create a new eDU for the call, with a new VDUID. The call then continues, and eContact Telephony will insert any new data into the new eDU; however, any data that was stored in the old eDU is lost.

To prevent recurrences of this condition, increase the value of **pseudo-ANI Timeout**.

Possible Complications

The problems discussed in this section are rarely encountered, and are restricted to certain configurations and particular phone switches. However, they can happen, and should be kept in mind.

- In some configurations, the following can occur. The Telephony Server sees the call and sends a TS.IncomingCall.event. However, the caller hangs up before the line to the VRU can register the call – so the VRU never sees the call and never runs the script.

  If the phone switch is of the kind that detects and reports on connections, the Telephony Server will not be informed of a connection in the above case. Therefore the Telephony Server will not send a TS.Connect.event to the VOX Server. Should the VOX Server next receive a VOX.newcall request from a *second* call, it will not respond incorrectly – by associating the TS.IncomingCall.event for the aborted call with the VOX.newcall request from the second call. It will not do this because it has never received a TS.Connect.event for the aborted call. When a new TS.IncomingCall.event is received from the second call, the VOX Server raises a *NoDeliver* informational alarm, discards the IncomingCall information for the aborted call, and handles the new call.

No problem occurred because the precaution was taken of requiring the VOX Server to wait for a TS.Connect.event before proceeding to respond to a telephone call. (See the discussion of "Wait for Connect," on page 33.)

- However, some phone switches (e.g., the Aspect phone switch) can *not* detect and report on connections. The Telephony Server knows about these switches and generates a *fake* TS.Connect.event for consistency. This fake event obviously does not confirm that the call was delivered to the VRU, and therefore does not provide the protection that was discussed in the preceding paragraphs.

  To provide protection in such cases, the **Wait For New Call** configuration parameter (see "Wait For New Call," on page 31) should be used. This sets a maximum time that the VOX Server will wait for a VOX.newcall request after receiving a TS.IncomingCall.event. After this time, the IncomingCall information is discarded, and the VOX Server begins afresh on that channel, again waiting for a TS.IncomingCall.event and VOX.newcall pair for the next call.

  The logic is that a VOX.newcall produced by the same telephone call that produced the TS.IncomingCall.event should arrive significantly faster than a VOX.newcall produced by a second call coming in after an aborted call. Therefore, if an appropriate maximum time is chosen, there should be no misassociation of an aborted call's TS.IncomingCall.event with a second call's VOX.newcall.

- However, choosing an appropriate maximum time can be tricky.
  - If it is too high, it is as if no limit had been set.
  - If it is too low, the VOX Server may prematurely discard the TS.IncomingCall.event, and when the delayed VOX.newcall for that call arrives it will find no matching TS.IncomingCall.event. It will wait for the *next* TS.IncomingCall.event, and then cause a misassociation. (Though a time limit can also be set to this waiting period, by using the **Wait For Incoming Call** configuration parameter – see "Wait For Incoming Call," on page 32.)

How to Find Out What Is Happening

If call misassociation occurs, the first place to look is at the alarm history. Whenever the VOX Server detects anything off in the ordering of events, it raises an alarm. Missing events raise the alarm *Miss*. Alarms that involve event errors typically have the following form:

```
xx seen:  Channel cc (state vduid)
```

where

xx	is **TS**, **TSCon**, or **Newcall** - whichever was the received event that gave the VOX Server the clue that something was wrong
cc	is the channel number
state	is one or more of the following symbols - indicating those events or responses that were already seen:
	TS - TS.IncomingCall.event seen
	TSCon - TS.Connect.event seen
	Script - Scripter response seen
	VRU - VOX.newcall seen
vduid	is the VDUID associated with the channel at this point (may be blank)

So the alarm

```
VOX.Miss      Newcall seen: Channel 10 (TS VRU TSCon Script 3034…002)
```

would indicate that a VOX.newcall just arrived at channel 10, and the channel state as known to the VOX Server already held a TS.IncomingCall.event, a VOX.newcall request, a TS.Connect.event, and a QWorkflow Designer response. A VOX.gone request (which would have cleared all four) was obviously missed.

A more serious alarm is

```
VOX.Miss      TS seen: Channel 12 (TS TSCon Script 3044…002)
```

indicating that a TS.IncomingCall.event arrived when a TS.IncomingCall.event, a TS.Connect.event, and a QWorkflow Designer response had already been received. A new call is arriving, and the VRU's VOX.newcall request from the previous call was never received. This may indicate a NoDeliver situation with a switch for which the Telephony Server generated a fake TS.Connect.event (see the previous section), or a VRU script that does not handle VOX.newcall properly.

Suggestions on Structuring VRU Scripts

The following should be treated as a flow chart or pseudocode. It must be adapted to each environment since each type of VRU has its own way of handling the mechanics of trapping hangups.

```
int vox_state; //create a flag that tells us what we’ve done with VOX
Server

Start_Of_Call://start here
vox_state = 0;//we’ve done nothing yet
Trap Hangup (hangup_seen)//if caller hangs up, jump to hangup_seen
Speak “Hello”//It’s OK, we’re being careful
vox_state = 1;//sending VOX.newcall...
Send “VOX.newcall”//send the newcall request, collect reply
....handle call....
Hangup Call//hangup call (don’t make channel ready yet)
sleep 10//wait for Hangup Trap to fire
//It didn’t FIRE?!
No Trap Hangup//cancel the trap
goto hangup_seen;//act like it fired

hangup_seen:
if (vox_state == 0)//did we send VOX.newcall??
        Send “VOX.newcall”//no, but we have to
if (vox_state < 2)//did we send VOX.gone?
        Send “VOX.gone”//no, but we must
        vox_state = 2
....wait for reply....
Enable Channel for next call
exit
```

Note: If some other part of the code sends a VOX.gone as part of other processing, it should set vox_state to 2 so that another VOX.gone will not be sent when within the hangup trap.

INDEX

Wait For New Call